AUSTRALIAN OS9 NEWSLETTER

EDITOR :
Gordon Bentzen
8 Odin Street
SUNNYBANK Qld 4109

(07) 345-5141

JULY 1990

ATTR Syntax: Attr filename [permissions] Usage : Examine or change the security permissions of a file Opts: -perm = turn off specified permission perm= turn on specified permission -a = inhibit rms : d - directory file s - n to owner w - write permit to ow or - read permit to public pw - te permit to public BACKUP Syntax ge : Copies all data from one d ead error occurs s = single rites BASIC09 Syntax : Basic0 ge BUILD Syntax: Build filenar s from standard input CHD S nge working directory to specifi > Usage: Change execution directory to specified path CMP Syntax: Cmp filename1 filename2 Usage : File comparison utility COBBLER Syntax: Cobbler devname Usage : Creates OS-9 bootstrap file from current boot CONFIG Syntax n disks COPY Syntax data from one fil E Syntax : Date [t Opts: t = specify ame> Usage : Check directory for wor sters -m = save of unused cluster nly -o = print <devname> {<devn : Del [-x] filenam s : -x = delete x: Deldir directo yntax: Dir [e] [x] the file names y x=print executi] Usage : Display s converted characters to standard output DSAVE Syntax : Dsave [-opts] [dev] [pathname] Usage : Generates procedure file to copy all files in a directory system Opts : -b make a system disk by using OS9boot if present -b=<path> = make system disk using path as source i = indent for directory levels l = do not process b makdir command o num K ECHO Syn tandard output ED oriented text edito s text error messages for given error numbers EX Syntax: ex <modname> Usage: Chain to the given module FORMAT Syntax : Format <devname> Usage : Initializes an OS-9 diskette Opts ; R - Ready L - Logical format only "disk name" 1/2 number of sides 'No of

AUSTRALIAN OS9 NEWSLETTER
Newsletter of the National OS9 User Group
Volume 4 Number 6

EDITOR : Gordon Bentzen

HELPERS : Bob Devries and Don Berrie

SUPPORT : Brisbane OS9 Level 2 User Group.

I'm trying to think but nothing happens!

What would you think of your favourite computer if it kept on saying this to you? The above phrase is a direct steal from a recently acquired public domain disk which included the archived file "play.ar". The PLAY command, written by Kevin Darling and modified by Brian C. White, will play most Mac and Amiga sound files. The command "play think" will result in that phrase coming from your monitor speaker. GREAT!

Yes, we do have more public domain programmes thanks to the efforts of Don Berrie, Bob Devries and the cooperation of members of overseas OS9 user groups. The files obtained are mostly in the AR archive format. The AR utility, by Carl Kreider, is loosely modelled on the archive utility in the Kernigan & Plauger book, "Software Tools" and its purpose is to gather together files into a common file to keep related files together. The files included in an archive typically include the executable module, source code .doc files, readme and any other related instruction files.

All the public domain library files are available by following the usual procedure. That is, mail your FORMATTED disks to address shown on the newsletter cover, include payment of our copy fee of $2 per disk (all disk formats 35ss, 40ds and 80ds attract the same charge) plus payment for return postage. We can also copy to three and half inch OS9 disks if required, or even MSDOS 360k disks (if you really want to do that).

We do incur costs in obtaining PD material, even though there may be no direct charge for the programmes, and the copy fee is designed to share these costs.

We do manage to deal with your requests for disk copies fairly promptly most of the time, however due to other commitments, a week or so may sometimes slip by. The easier you make it for us, the quicker we can return mail your disks. Please include a self addressed label, postage stamps for return mail and use a mailer or package which we can reuse for return. This means that we can repack the copied disks back in your mailer, stick on your address label and stamps, and pop it in a mail box.

Most of the PD is stored on 80 track double sided disks, and if we can simply do a backup, this also speeds things up.

Now the question on your lips is; What PD programmes do we have and how many disks are needed? Very good question! The problem of indexing and listing the programmes in a meaningful way is a problem to which we do not have an ideal answer at this point.

The newly acquired material was sourced from the European OS9 User Group and came on twenty-eight (28) 40tr DS plus three (3) 80tr DS disks. There appears not to be any particular order to the type of file, and files are mostly in the root directory of each disk. We will need a period of grace to sort through these disks if particular programme types are sought. We are, however, most grateful for the material.

We also have a number of disks from the U.S. OS9 User Group in the original format, Disk1 through Disk42 (with a couple missing). These were issued on 35tr single sided disks, and are now in the library on 80 track disks under a directory for each issue, eleven 80tr disks in all. The same files have also been sorted into categories, eg. Word Processing, Communication etc., nine 80tr disks in all. These disks are now a few years old and do not contain any programmes specifically for level 2. Some OS9 68k is included.

All in all there is enough material to keep anybody busy for months even if it is somewhat of a lucky dip at the moment. We will include some comment and review of the more interesting programmes in our future newsletters for those who wish to be selective.

"Things are certainly starting to happen in the OS9 World!" That was the opening statement of last month's editorial which dealt with new OS9 machines. The new OS9 programmes being advertised overseas together with the increasing volumes of PD software certainly add weight to that statement.

To keep the ball rolling we can add our bit, even if it seems but a drop in the ocean. We desperately need Australian produced commercial programmes and public domain material. The aim of the National OS9 User Group is to promote the use of OS9 on a non-profit basis, so please assist us by sharing any programme of your own which you are prepared to release as public domain or shareware. Many of our members want to see more Basic09 programmes. Can you help?

Also to keep the ball rolling, and to maintain supply of PD software from overseas sources, we do need to send something in return. Your little Basic09 programme or utility could spread throughout Australia, Europe and America.

SUBSCRIPTIONS:- Membership of the National OS9 User Group is still only $18 per annum. Sincere thanks to those who have already sent a cheque for membership renewal.

We maintain a common expiry date of end August each year, and we do need your continued support to avoid a rise in fees.

PLEASE, PLEASE, PLEASE take the time to include a short note with your subscription renewal and tell us what you would like to see in this newsletter; also tell us what you don't like. Please make your cheque payable to "National OS9 User Group"

JUST A REMINDER:- We do not accept PAID advertising (subscriptions must cover all costs) however, we will include ads from members for locally produced hardware or software, and ads (unpaid) from commercial enterprises, only if we consider such an ad would be "a service to members".

I'm trying to think but STILL nothing happens! Cheers, Gordon.

oooooooooOOOOOOOOOooooooooo

CC3GO AND SHELLPLUS

The modified shell programme, Shellplus (Version 2.xx and onwards) gives us all kinds of nifty opportunities to get better productivity from our CoCo OS9 systems. One of those nice things that shellplus offers is the ability for the user to set a search path for executable files. This allows the system to examine a number of directories, not just the current execution directory, for executable files.

Similarly, the user has the opportunity to use a prompt other than the standard OS9 system prompt. It is possible for the user to actually modify the shellplus executable file, and hardcode the desired prompt within the shellplus code. However, we are all aware of the problems caused by hard coded device names, parameters etc, and of course they all have the drawback that they then cannot be easily changed.

Because of the way that OS9 Level 2 for the CoCo3 forks the startup file from it's specific sysgo module (called cc3go), unless you actually hard coded the paths into that module, you would not normally be able to have any paths or custom prompts set, on your /Term device.

To attempt to get around this limitation, we are now able to offer, as part of our public domain library, an archived set of files containing a replacement module for cc3go, written by Bob Devries and Don Berrie. This new module tries to open a one-line file called /dd/SYS/config.os9, and expects to find in that file the parameters required for the shell which it opens on the default device (normally /Term).

Included in the archive is also a sample config.os9 file as an example. If the file is cannot be found, an error message is printed. Similarly, if the system cannot read the file, an error message to indicate this problem is also printed. In either case, the system then simply uses the standard default parameter i=/1 as in the original cc3go module.

For those users who have a Burke & Burke hard drive with the XT-ROM installed in the interface, and wish to use the alternate boot feature of that system, we have also included a module called cc3go.alt. This module looks for files named /dd/SYS/config.alt, altstart, and altshell rather than /dd/SYS/config.os9, startup and shell. This will allow you to have two totally different bootfiles on your hard drive.

The source code, fully commented, for both modules is also included.

The usual National OS9 Users Group copying conditions apply. Any comments or requests for further information should be directed to the authors:

Bob Devries
(07) 3727816

Don Berrie
(07) 375 3236

oooooooooo0000000000oooooooooo

HOW COME MY BOOT PROCESS TAKES SO LONG?

Ever wanted to know what is happening while you are sitting there looking at that green VDG screen with OS9 BOOT written in the middle of it? Well read on.

This information comes from the OS9 "super guru" Kevin Darling, and is reproduced with thanks.

The OS9 Boot sequence is as follows:

The DOS routine in RSDOS, or some other type of custom ROM, points to a track (normally Track 34) on a disk drive, and reads that track, which has REL, BOOT, and OS9P1.

REL sets up the main GIME registers and the system crash/reset vectors, and then bypasses Boot to execute the OS9P1 module.

OS9p1 sets up Direct Page variables, F$.... system calls, finds free RAM, and scans RAM for module headers. It then inserts found modules (at this stage Rel, Boot and OS9p1) into the module directory, and looks for the Init module. OS9p1 can't find Init, because it has not as yet been loaded from the disk, so it does an internal F$Boot. F$Boot calls the Boot module, which *hopefully* loads the bootfile from disk. F$Boot is not particularly smart, and amongst other things, requires that the bootfile is in one single contiguous block. OS9p1 then links to OS9p2 and jumps to it.

OS9p2 installs its F$calls, and then looks in Init for its default execution and data directories, and attempts to change to them using I$ChgDir. I$ChgDir fails because no I$calls are as yet installed, so OS9p2, by default, finds IOMan and initializes it, and I$ChgDir is tried again. OS9p2 opens a path to the default device (usually /term, but hard coded into the Init module.)

OS9p3 is then looked for, and, if found, it is called as a subroutine. (If OS9p4 is present it is called from OS9p3 ... &etc)

Once the user defined extensions to OS9 (OS9p3, OS9p4 ....) have been called and executed, OS9p2 then checks Init for the initial process name (CC3Go), and executes it using an F$Fork call. It then does an F$NProc call which doesn't return.

CC3Go prints the "OS9 LEVEL 2 .... " signon message, and sets the default time (setime causes OS9p2 to find the Clock module and init it at this time), which is also hard coded into the CC3Go module.

CC36o attempts to change directories to /h0. If successful, it modifies the system process descriptor to also use /h0 (for calling in grfdrv later, etc). CC36o tries to fork "shell startup" and waits until the startup process has completed. CC36o then tries to fork "autoex" and if found will sit and wait until the execution of this programme terminates.

CC36o then chains (which means that it terminates itself, and starts a new process) to the Shell on the default window defined in the Init module. (This shell is not started from the startup procedure, hence only the preset shell parameters, which are hardcoded into CC36o, are set for this shell).

Then, you finally get control of the system. Whew!

oooooooooo0000000000oooooooooo

BUILDING A BOOT DISK.....Part 2

If you followed part 1 in last month's News Letter, you should have a system disk that is configured to take advantage of the maximum capacity of your disk drives.

As promised I will explain how to build a customized system disk using only the modules supplied on the original Level 2 system disk....So Boot up your system. Make sure you are using a copy of your system disk in drive /d0. We need to create a MODULES directory on this disk and copy into it the modules needed for your System Disk.

```
Makdir /d0/MODULES
```

By now you should have decided which modules you require in your Boot file, if not, make yourself a list now. Place a backup copy of your Boot-Config-Basic09 disk in drive /d1 and copy the files you require into the modules directory on the disk in /d0.

```
chd /d0/modules
copy /d1/modules/CC3Disk.dr CC3Disk
copy /d1/modules/clock.50hz Clock
copy /d1/modules/d0_40d.dd d0
copy /d1/modules/dd_40d.dd dd

etc.....
```

Until you have all the modules required to create your new boot disk. It is a bit tedious, but once done, you shouldn't have to do it again. You can add, delete or change modules as your system requirements change. You will have noticed that I left the extensions off the files I copied, they are not needed.

While I did state that we wouldn't be using anything that was not on the original system disks supplied by Tandy, should you have the "save" command from Level 1 or from the Development System disks, then to save some time I would suggest you save the modules from memory that you require, especially those modules that have been patched.

```
chd /d0/modules
save d0
save d1
save dd

etc.....
```

The modules directory can contain as many files as you wish, it is not restricted to only the files required on a particular Boot Disk. It will be a permanent source for creating different Boot Disks when required.

Now we need to create two files, the first is to be a file called "bootlist", which will contain a list of the modules to be included in the new os9boot file. The other is the procedure file that will create the new OS9Boot file, We will call this file "new_boot". These files can be created by using the Build command or with Edit, or as I do with a word processor.

The list of module names in "bootlist" should be in the order they are wanted in the Boot file.

Bootlist.....

```
OS9p2
IOMan
RBF
CC3Disk
D0
D1
DD

etc....
```

Once you have created this bootlist in the data directory of /d0, the next thing is to create the procedure file new_boot which must also reside in the data directory of /d0.

Procedure New_Boot

```
t
tmode .1 -pause
chd /d0/modules
os9gen /d1 </d0/bootlist
chd /d0
dsave /d0 /d1 ! shell
tmode .1 pause
-t
```

Now with a formatted disk in /d1 type new_boot then press enter and a new bootdisk will be created that includes only those files listed in Bootlist. The two lines Chd /d0 and dsave /d0 /d1 ! shell can be removed if you don't want the directories and files on /d0 copied to the new boot disk. You can copy these files from any source you choose, bearing in mind that shell and GrfDrv must be in the commands directory otherwise the boot will fail. Once the boot disk is completed, you can boot up with the new system disk and then make the necessary patches etc. according to last months' article.

During the writing of this article I have come to realise just how far we have progressed since OS9 Level 2 was first released. An overview of the modifications being used by the Brisbane User Group Members might be in order.

Regards Rob Unsworth.

oooooooooo0000000000oooooooooo

Check Book Programme.

Programme by Dale L. Puckett, comments by Bob Devries.

Here's a short Basic09 programme to assist in balancing your checkbook. The programme came to us by courtesy of the European OS9 Usergroup, and originated from Compu-Serve Information System. The source is very short and easy to understand. It requires no special software (except Basic09 of course) and should work on both level one and level two OS9 systems.

Bob Devries.

```
PROCEDURE Checkbook
 0000      (* Something to Help balance your checkbook *)
 002E
 002F      DIM answer,clearcode:STRING[1]
 003F      DIM balance,service_charges:REAL
 004A      DIM total_checks,outstanding_deposits:REAL
```

```
0055    DIM number_of_checks,number_of_deposits:INTEGER
0060
0061    clearcode:=CHR$($0C)
006A
006B    PRINT clearcode
0070
0071    PRINT
0073    INPUT "What was the balance on the statement? ",balance
00A2    INPUT "What was the total of all service charges? ",service_charges
00D5
00D6    PRINT clearcode
00DB
00DC    number_of_checks:=-1
00E4    total_checks:=0
00EC    amount:=0
00F4
00F5    RUN getchecks(number_of_checks,total_checks,amount)
0109
010A    PRINT clearcode
010F
0110    number_of_deposits:=-1
0118    outstanding_deposits:=0
0120    amount:=0
0128
0129    RUN getdeposits(number_of_deposits,outstanding_deposits,amount)
013D
013E    PRINT clearcode
0143
0144    PRINT
0146    PRINT "Your final balance should be  $";
016A    PRINT USING "R8.2^",balance+outstanding_deposits-(total_checks+service_charges)
0184    PRINT  \ PRINT  \ PRINT
018A    PRINT "From your statement: "
01A3    PRINT
01A5    PRINT "A balance of  $";
01B9    PRINT USING "R8.2^",balance;
01C8    PRINT "minus a service charge of  $";
01E9    PRINT USING "r9.2^",service_charges;
01F8    PRINT " =  $";
0202    PRINT USING "r8.2^",balance-service_charges
0214    PRINT
0216    PRINT "You had "; number_of_checks; " outstanding check(s), totalling  $";
024D    PRINT USING "r8.2^",total_checks
025B    PRINT "and "; number_of_deposits;
0268    PRINT " deposit(s) outstanding, totalling:   $";
0294    PRINT USING "r8.2^",outstanding_deposits
02A2    PRINT
02A4
02A5
02A6    INPUT "Would you like to balance another statement: (Y)es or (N)o? ",answer
02EA
02EB    IF answer="Y" OR answer="y" THEN
0300      RUN checkbook
0304    ELSE
0308      PRINT
030A      PRINT "Hope we were able to help you with your headache."
033F    ENDIF
0341
```

```
0342      END
0344

PROCEDURE getchecks
0000      PARAM number_of_checks:INTEGER
0007      PARAM total_checks,amount:REAL
0012
0013      PRINT
0015      PRINT "Now we need to have you list the amount of each check"
004E      PRINT "that was not listed on the bank's statement."
007E      PRINT
0080      PRINT "When you have listed all the checks enter a value of zero."
00BE      PRINT
00C0
00C1
00C2      REPEAT
00C4        INPUT "Amount of check? ",amount
00DD        number_of_checks:=number_of_checks+1
00E8        total_checks:=total_checks+amount
00F4      UNTIL amount=0
0100
0101      END

PROCEDURE getdeposits
0000      PARAM number_of_deposits:INTEGER
0007      PARAM outstanding_deposits,amount:REAL
0012
0013      PRINT
0015      PRINT "You must now list each deposit that did not show up"
004C      PRINT "on the bank statement."
0066      PRINT
0068
0069      PRINT "Enter a zero when all your deposits have been entered."
00A3      PRINT
00A5
00A6
00A7      REPEAT
00A9        INPUT "Amount of deposit? ",amount
00C4        number_of_deposits:=number_of_deposits+1
00CF        outstanding_deposits:=outstanding_deposits+amount
00DB      UNTIL amount=0
00E7      END
```

oooooooooo0000000000oooooooooo

Window Directory

programme by Kevin Darling, text by Bob Devries.

More Basic09 you say? Well OK. Here's a little utility written by Kevin Darling, and obtained by me via the European OS9 community from the Compu-Serve Information System. The programme displays a directory of the currently opened or INIZ'd windows in your system. It tells you the entry number, the window type, which block the screen ram uses, the current working area, and the sixteen palette register contents. I have found it useful to find out what colours are set in each window. The programme is not very large, and uses various OS9 system calls via the 'SysCall' utility.

Bob Devries.

```
PROCEDURE WDir
0000      REM Window Directory
0013      REM Copyright by Kevin Darling Aug 87
0037      REM For Each Window WDir Shows:
0055      REM Window table entry #
006C      REM Window type
007A      REM Block numbers used, with offset into block if 1 block
00B2      REM Backlink window table # if overlay, and overlay buffer
00EB      REM block begin number.
0101      REM Window begin and size (col,row) numbers
012B      REM Window current cwarea being and size
0152      REM Palette values (0-15) for that screen
017A      REM Really should be rewritten with recursive calls to
01AF      REM show all parent backlinks of multiple overlay W's.
01E4      REM Set Up Vars and Define Constants:
0208      BASE 0
020A      DIM temp:BYTE
0211      DIM w:INTEGER
0218      DIM WNum:BYTE
021F      DIM sctab:INTEGER
0226      TYPE register=CC:BYTE; D:INTEGER; DP:BYTE; X,Y,U:INTEGER
024B      DIM reg:register
0254      DIM WE(64):BYTE
0260      DIM SysPrc(512):BYTE
026C      DIM D_DevTbl:INTEGER
0273      TYPE table=V_Driv,V_Stat,V_Desc,V_FMgr:INTEGER; V_Usrs:BYTE
0290      DIM devtable(32):table
029E      DIM DrvName:STRING[5]
02AA      DIM DevName:STRING[4]
02B6      DIM M_Name:INTEGER
02BD      DIM cc3io:STRING[5]
02C9      DIM SC(32):BYTE
02D5      DIM typ$(9)
02DF      DIM F_GPrDsc:BYTE
02E6      DIM F_CpyMem:BYTE
02ED      cc3io="CC3I"+CHR$($CF)
02FD      F_GPrDsc=$18
0305      F_CpyMem=$1B
0300      FOR n=0 TO 8
031F        READ typ$(n)
0329      NEXT n
0334      DATA "Same Screen"
0346      DATA "40 Col Text"
0358      DATA "80 Col Text"
036A      DATA "Bad"
0374      DATA "Bad"
037E      DATA "640 Two Color"
0392      DATA "320 Four Color"
03A7      DATA "640 Four Color"
03BC      DATA "320 Sixtn Color"
03D2      REM Get SysPrc Descriptor:
03EB      reg.D=256
03F7      reg.X=ADDR(SysPrc)
0405      RUN Syscall(F_GPrDsc,reg)
0414      REM Get Offset in System Map to Device Tables:
0441      destination=ADDR(D_DevTbl)
044C      count=2
0454      offset=$80
```

```
0450      GOSUB 1000
0461      REM Now Use that Info to Get the Device Table Itself:
0495      destination=ADDR(devtable)
04A0      count=SIZE(devtable)
04AB      offset=D_DevTbl
04B4      GOSUB 1000
04B8      REM Main Loop:
04C5      REM Check Each Device Entry for CC3IO Driver.
04F1      REM If it is, then Get the Window Entry # From Static Storage.
052E      REM (unless type <> window, then treat as VDG)
055B      REM (and skip deleted entries)
0578      FOR DE=0 TO 31
058A        REM Check for Entry in Use:
05A4        IF devtable(DE).V_Usrs<>0 THEN
05B7          REM Get Driver Name Offset In Descriptor:
05DF          destination=ADDR(M_Name)
05EA          count=2
05F2          offset=devtable(DE).V_Driv+4
0605          GOSUB 1000
0609          REM And Then the Name Itself:
0625          destination=ADDR(DrvName)
0630          offset=M_Name+devtable(DE).V_Driv
0644          count=5
064C          GOSUB 1000
0650          IF DrvName=cc3io THEN
065D            GOSUB 500
0661          ENDIF
0663        ENDIF
0665      NEXT DE
0670      PRINT "---------------------------------------"
069B      END
069D 500  REM ***********************************
06C3      REM Print Window Entry This Device:
06E5      destination=ADDR(M_Name)
06F0      offset=devtable(DE).V_Desc+4
0703      count=2
070B      GOSUB 1000
070F      destination=ADDR(DevName)
071A      offset=M_Name+devtable(DE).V_Desc
072E      count=4
0736      GOSUB 1000
073A      PRINT "---------------------------------- ";
0767      n$=""
076E      FOR c=1 TO 4
0780        t$=MID$(DevName,c,1)
078F      EXITIF ASC(t$)>128 THEN
079C        t$=CHR$(ASC(t$)-128)
07A9        n$=n$+t$
07B5      ENDEXIT
07B9        n$=n$+t$
07C5      NEXT c
07D0      PRINT n$
07D5      REM Check Device Static Memory For Type:
07FC      destination=ADDR(temp)
0807      offset=$06+devtable(DE).V_Stat
0818      count=1
0823      GOSUB 1000
0827      IF LAND($80,temp)=0 THEN
```

```
0837        PRINT "VDG Screen"
0845        RETURN
0847      ENDIF
0849      REM Else Get Window Entry# From Static Mem:
0873      destination=ADDR(WNum)
087E      offset=$35+devtable(DE).V_Stat
0892      count=1
089A      GOSUB 1000
089E      w=WNum
08A6      offset=$1280+w*64
08B6      count=64
08BE      destination=ADDR(WE)
08C9      GOSUB 1000
08CD      sctab=WE(0)*256+WE(1)
08E1      PRINT "Entry# "; w
08F0      IF LAND(sctab,$FF00)=$FF00 THEN
0901        IF LAND(sctab,$FF)=$FE THEN
0912          PRINT "Iniz'd"
091C        ELSE
0920          PRINT "!! DEINIZ or DWSET THIS DEVICE !!"
0945        ENDIF
0947      ELSE
094B        offset=sctab
0954        count=32
095C        destination=ADDR(SC)
0967        GOSUB 1000
096B        PRINT "Window Type :";
097D        wtype=SC(0)
0988        IF wtype=$FF THEN
0996          PRINT "Iniz'd"
09A0          RETURN
09A2        ENDIF
09A4        IF wtype>$80 THEN
09B2          wtype=$87-wtype
09BF        ELSE
09C3          wtype=wtype+4
09CF        ENDIF
09D1        PRINT USING "i3>",wtype;
09DE        PRINT "        "; typ$(wtype)
09F2        REM PRINT "Screen Table: ";
0A0D        REM PRINT USING "h4",sctab
0A26        PRINT "Block Number: ";
0A39        PRINT USING "h2",SC(1);
0A47        IF wtype>2 THEN
0A54          PRINT "-";
0A5A          IF wtype=5 OR wtype=6 THEN
0A6F            PRINT USING "h2",SC(1)+1
0A7F          ENDIF
0A81          IF wtype=7 OR wtype=8 THEN
0A96            PRINT USING "h2",SC(1)+3
0AA6          ENDIF
0AA8        ELSE
0AAC          PRINT "        Offset: ";
0AC1          PRINT USING "2(h2)",WE($34)-$80; WE($35)
0AD0        ENDIF
0ADF        IF WE(2)<>$FF THEN
0AEE          PRINT "Parent Entry: ";  \ PRINT USING "i2>",WE(2);
0B10          PRINT "        Overlay ";
```

```
0B25          PRINT USING "h2",WE($21)
0B33        ENDIF
0B35        PRINT "Window Start: "; WE($36); ","; WE($37); "    ";
0B61        PRINT "   Size : "; WE($38); ","; WE($39)
0B81        PRINT "CWArea Start: "; WE($05); ","; WE($06);
0BA6        PRINT "       Size : "; WE($07); ","; WE($08)
0BCA        FOR prn=0 TO 7
0BDC          PRINT USING "h2",SC(16+prn);
0BF0          PRINT " ";
0BF6        NEXT prn
0C01        PRINT
0C03        FOR prn=8 TO 15
0C15          PRINT USING "h2",SC(16+prn);
0C29          PRINT " ";
0C2F        NEXT prn
0C3A        PRINT
0C3C      ENDIF
0C3E      RETURN
0C40 1000 REM Copy count at offset in datimage to destination:
0C76      reg.D=ADDR(SysPrc)+$40
0C88      reg.Y=count
0C95      reg.X=offset
0CA2      reg.U=destination
0CAF      RUN Syscall(F_CpyMem,reg)
0CBE      IF LAND(reg.CC,$01)<>0 THEN
0CD1        PRINT "Error "
0CDB        PAUSE
0CDD        END
0CDF      ENDIF
0CE1      RETURN
0CE3
```

oooooooooo0000000000oooooooooo

COCO-LINK Magazine: Just a reminder for those interested in a wider section of CoCo systems. CoCo-Link is produced in South Australia and is well worth your support. Contact the Editor, Robbie Dalzell, on (08) 386 1647 or write;

CoCo-Link
31 Nedland Crescent,
Pt. Noarlunga South,
South Australia 5167

oooooooooo0000000000oooooooooo